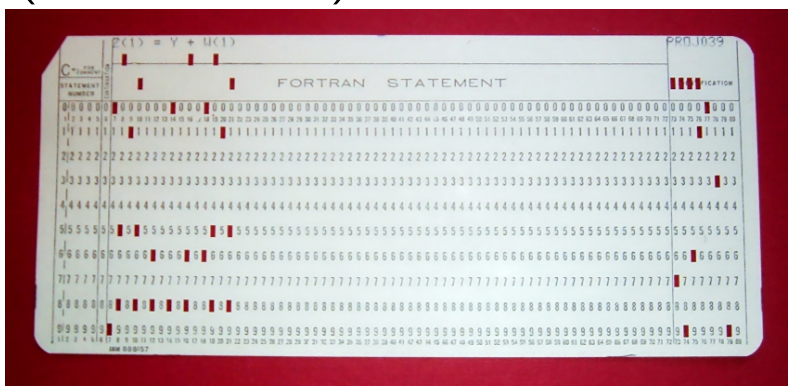
Metody numeryczne

Wykład 0 - Programowanie w Octave

Historia

- lata 50 XX w. – Fortran - kompilator wysokiego poziomu do obliczeń naukowych i inżynierskich (John Backus)

https://en.wikipedia.org/wiki/Fortran

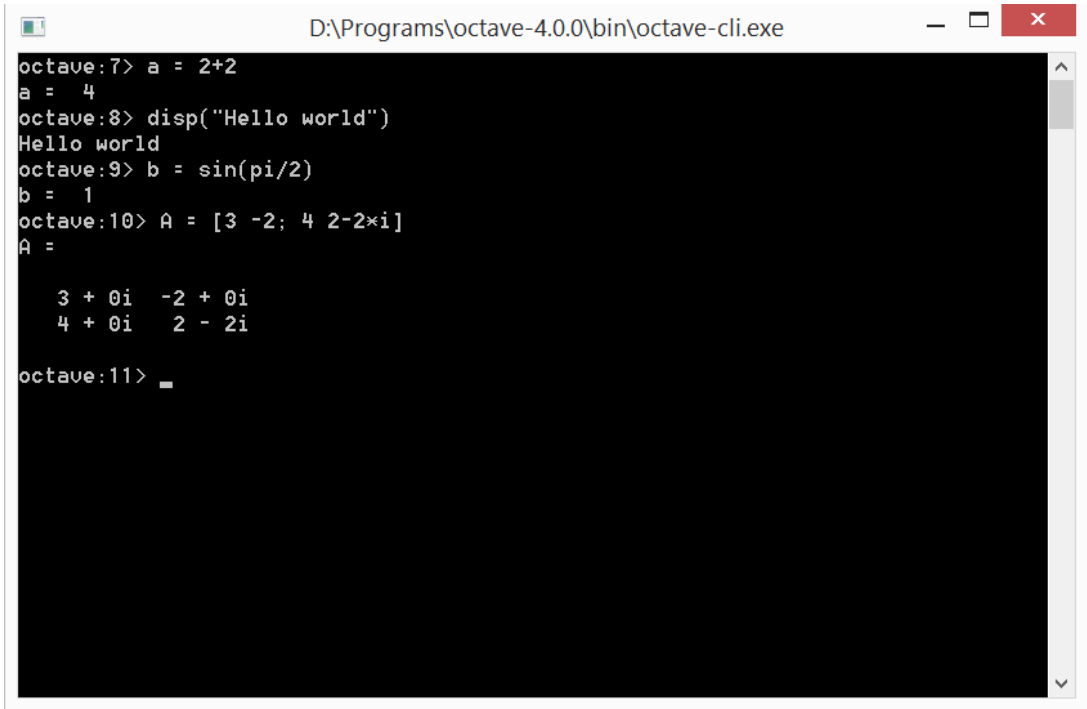


https://en.wikipedia.org/wiki/Punched˙card

- lata 70 XX w. – Linspack, Eispack – biblioteki do obliczeń macierzowych dla Fortranu
- 1980 - darmowa aplikacja do korzystania z Linspack'a i Eispack'a bez znajomości Fortranu (Cleve Moller)
- 1985 - pierwsza wersja komercyjnego programu Matlab (C)
- 1993/94 – pierwsza wersja Octave (John W. Eaton) "darmowy Matlab"
- 2015 – Octave 4.0.0

Cechy środowiska Octave/Matlab

- środowisko obliczeniowe wraz z językiem programowania wysokiego poziomu
- (zasadniczo) język interpretowany proceduralny (są obiekty ale lepiej ich nie używać, da się kompilować ale nie będziemy tego robić)
- duże podobieństwo składni (możliwość napisania kodu przenośnego Matlab $\leftrightarrow$ Octave)
- tryb interaktywny (konsola) oraz wsadowy (skryptowy)



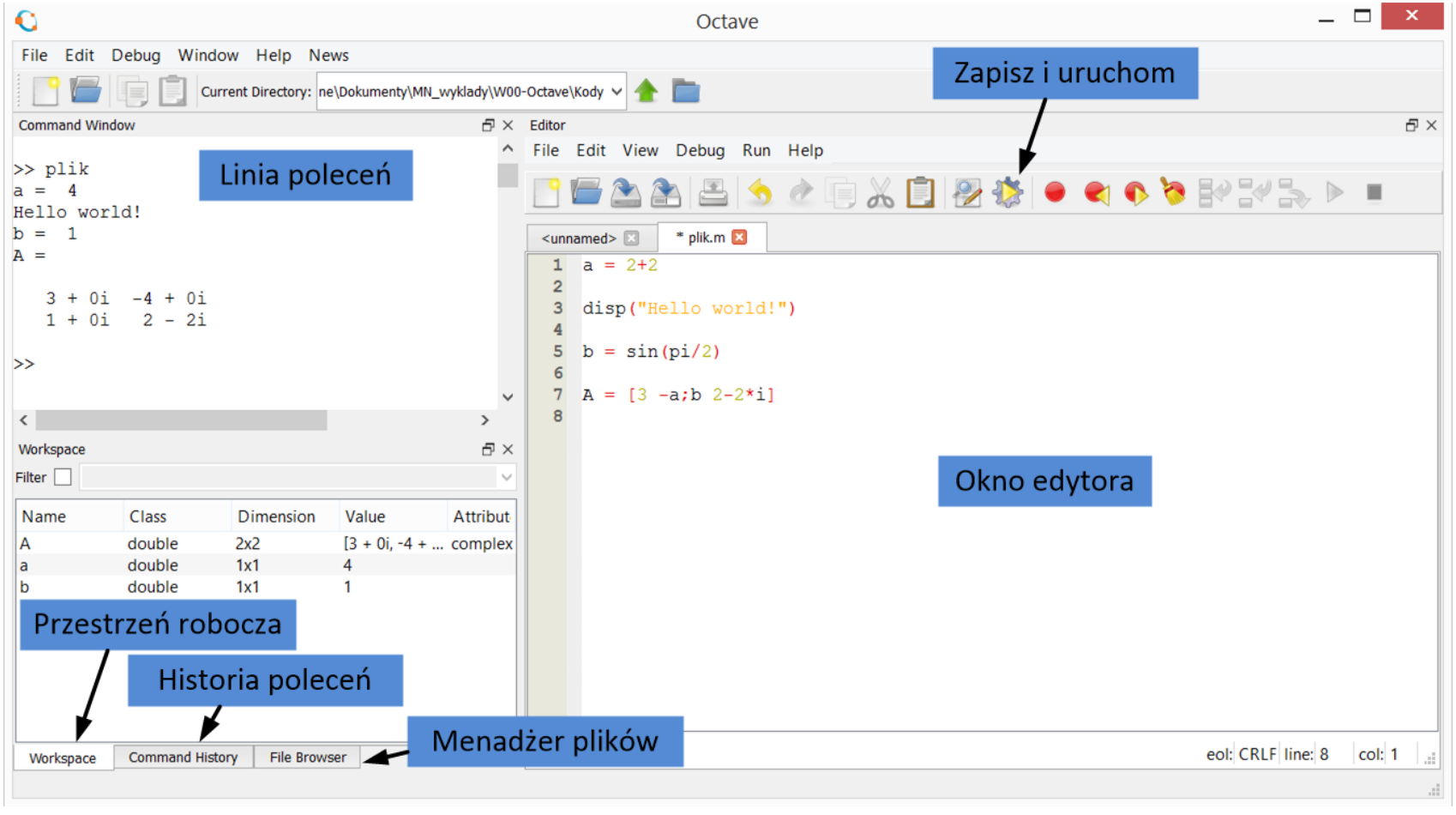
Cechy środowisk Octave/Matlab

- składnia wzorowana na C ale:
  - brak wskaźników, referencji
  - brak deklaracji typu zmiennej (typ domyślny: macierz wypełniona liczbami zmiennoprzecinkowymi podwójnej precyzji)
  - argumenty przekazywane do funkcji przez wartość (ale dzieje się tak tylko wtedy gdy argument jest modyfikowany wewnątrz funkcji!)

```
x = rand(1000) ;
y = f(x) ;
```

- tryb wsadowy: skrypty/funkcje w m-plikach

Octave GUI (> Octave 3.8)



Tryb interaktywny

- "kalkulator"

```
>> 2+4
ans =  6
```

- wartość wyrażenia można przypisać do zmiennej (domyślna zmienna to ans)

```
>> a = 2*3
a =  6
>> b = sqrt(2) + sin(a)
b =  1.1348
```

- średnik na końcu powoduje nie wyświetlanie wyniku polecenia

```
>> b = 3 ;
```

- tabulacja – wyświetlenie "podpowiedzi"

```
>> cart2 %(Tab)
cart2pol  cart2sph
```

- strzałki w górę/w dół - przywoływanie poprzednich poleceń
- help

```
>> cart2 %(Tab)
cart2pol  cart2sph
```

Tryb wsadowy: skrypty/funkcje

- kod programu zapisywany w pliku tekstowym z rozszerzeniem .m
- tworzenie pliku
- zmiana katalogu bieżącego/dodanie katalogu do ścieżki przeszukiwań

```
>> cd('/sciezka/do/katalogu')
>> path
>> addpath('/sciezka/do/katalogu')
```

  * w oknie edytora F5 → 'Add Directory to Load Path'/'Change Directory'

- wykonanie pliku →
  - w linii poleceń

```
>> nazwa_pliku      % bez rozszerzenia!
```

  - w edytorze → F5

- brak deklaracji zmiennych i alokacji pamięci

```
a = 10 ;
s = "Napis" ;
S = 'to jest inny napis' ;
```

- nadpisywanie zmiennych innym typem

```
s = 200/3 ;
```

- formaty wyświetlania wartości liczbowych

```
>> format long ;
>> s
s =  66.666666666667
>> format bit ;
>> s
s = 0100000001010000101010101010101010101010101010101010101010101011
>> format short e ;
>> s
s =    6.6667e+01
>> format long e ;
>> s
s =    6.66666666666667e+01
>> format short ;
>> s
s =  66.667
```

i in.

- jeszcze inne sposoby wypisywania zmiennych

```
>> printf('%4.3f \n', s)
66.667
>> printf('%1.6f \n', s)
66.666667
>> printf('%s \n', S )
Inny napis
>> disp('')

>> disp(s)
 66.667
>> disp(S)
Inny napis
```

- różne inne przydatne funkcje

Funkcja	Opis
clc	czyszczenie okna poleceń
clear A B	usunięcie z pamięci podręcznej zmiennej A, B, itd.
clear all	usunięcie całej pamięci podręcznej
who	wyświetla nazwy zmiennych lokalnych przechowywanych w pamięci podręcznej
whos	to samo co powyżej + dodatkowe informacje na temat każdej zmiennej
tic	uruchomienie "stopera"
toc	wyłączenie "stopera" i wyświetlenie czasu od jego uruchomienia (w sekundach)
pwd	wyświetlenie aktualnego katalogu roboczego
cd	przejście do katalogu, który staje się katalogiem roboczym
dir	wyświetlenie zawartości aktualnego katalogu roboczego
ls	wyświetlenie zawartości aktualnego katalogu roboczego
quit	wyjście z programu
exit	wyjście z programu

c.d.

- różne inne przydatne funkcje

Funkcja	Opis
save('plik1')	zapis całej pamięci podręcznej do pliku plik1
save('plik1', 'A', 'B')	zapis do pliku plik1 wybranych zmiennych
load('plik1')	odczyt zmiennych zapisanych w pliku plik1 i umieszczenie ich w pamięci podręcznej
more on	"stronicowanie" tekstów wysyłanych na ekrań (do okna linii poleceń)
more off	wyłączenie "stronicowania"

Typy danych – macierze

Tworzenie macierzy:

- wymienienie elementów

```
A = [1 2 3; 7 8 9]
A = [1,2,3; 7,8,9]
A = [1 2 3
     7,8,9]
```

- generowanie elementów

  – operator ":"

```
>> w = 0:3:6
w =
   0   3   6

>> w = 0:6
w =
   0   1   2   3   4   5   6

>> w = 2:-0.5:1
w =
    2.00000    1.50000    1.00000
```

- sklejanie innych z macierzy

```
>> A = 1:4 ;
>> B = 1:5 ;
>> C = 0 ;
>> D = [A, C; B]
D =
   1   2   3   4   0
   1   2   3   4   5
```

Funkcje wspomagające generowanie macierzy:

Funkcja	Opis
eye(n) eye(n,m)	macierz jednostkowa o wymiarach $n \times n$ lub $n \times m$
ones(n) ones(n,m)	macierz wypełniona jedynkami
rand(n) rand(n,m)	macierz wypełniona liczbami pseudolosowymi o rozkładzie jednostajnym na [0,1]
zeros(n) zeros(n,m)	macierz wypełniona zerami

```
>> D=eye(3,4)
D =

Diagonal Matrix

   1   0   0   0
   0   1   0   0
   0   0   1   0

>> w = 5 * ones(1,6)
w =

   5   5   5   5   5   5

>> rand(3)
ans =

   0.37275   0.27255   0.51789
   0.83291   0.18262   0.27950
   0.40963   0.58793   0.33595
```

Odwoływanie się do elementów macierzy

```
>> V = [ [9:-1:7; 1:3], zeros(2,1);
         1 1 1 1]
V =

    9    8    7    0
    1    2    3    0
    1    1    1    1

>> V(1)  % a nie V(0)
ans =  9

>> V(1,3)
ans =  7

>> V(3,1)
ans =  1

>> V(:,2)
ans =
    8
    2
    1

>> V(2,:)
ans =
    1    2    3    0

>> V(2, 1:3)
ans =
    1    2    3

>> V(2, [1 3 4])
ans =
    1    3    0

>> V([1 3], [1 3 4])
ans =
    9    7    0
    1    1    1

>> V(7)
ans =  7

>> V(:)
ans =
    9
    1
    1
    8
    2
    1
    7
    3
    1
    0
    0
    1
```

Usuwanie elementów macierzy

```
>> V(:,4) = []
V =
    9    8    7
    1    2    3
    1    1    1

>> V(2,:) = []
V =
    9    8    7
    1    1    1
```

Typy danych – stringi

Stringi to tak naprawdę również macierze!

```
>> s1 = "laczenie"
s1 = laczenie
>> s2 = 'kilku'
s2 = kilku
>> s3 = ["stringow"]
s3 = stringow
>> s4 = ["a tu " s1 ' ' s2 ' ' s33]
s4 = a tu laczenie kilku stringow
>> disp(s4)
a tu laczenie kilku stringow
>> s1(3)
ans = c
```

Operatory macierzowe - operatory algebraiczne

```
>> A = [1 0 0;
        2 3 -1;
        0 7 2] ;

>> B = [1 0 3;
       -1 5 2;
        2 2 2]   ;

>> A+B
ans =
   2    0    3
   1    8    1
   2    9    4

>> A-B
ans =
   0    0   -3
   3   -2   -3
  -2    5    0

>> A+2 % A + 2*ones(size
      (A))
ans =
   3    2    2
   4    5    1
   2    9    4
```

```
>> 2*A
ans =
     2     0     0
     4     6    -2
     0    14     4

>> 2.*A
ans =
     2     0     0
     4     6    -2
     0    14     4
```

```
% Mnozenie macierzowe
>> MM1 = A*B
MM1 =
     1     0     3
    -3    13    10
    -3    39    18

>> MM2 = B*A
MM2 =
     1    21     6
     9    29    -1
     6    20     2

% Mnozenie tablicowe
>> MT1 = A.*B
MT1 =
     1     0     0
    -2    15    -2
     0    14     4

>> MT2 = B.*A
MT2 =
     1     0     0
    -2    15    -2
     0    14     4
```

Operatory macierzowe - operatory algebraiczne

```
A = [1 0 0;
     2 3 -1;
     0 7 2] ;

B = [1 0 3;
    -1 5 2;
     2 2 2]  ;
```

```
% Dzielenie macierzowe
>> DM1 = A/B
DM1 =
   -0.20000   -0.20000    0.50000
   -1.40000   -0.06667    1.66667
   -0.60000    1.06667    0.83333

>> DM2 = A\B
DM2 =
    1.00000    0.00000    3.00000
   -0.30769    0.92308   -0.46154
    2.07692   -2.23077    2.61538

% Dzielenie tablicowe
>> DT1 = A./B
DT1 =
    1.00000         NaN    0.00000
   -2.00000    0.60000   -0.50000
    0.00000    3.50000    1.00000

>> DT2 = A.\B
DT2 =
    1.00000         NaN         Inf
   -0.50000    1.66667   -2.00000
        Inf    0.28571    1.00000
```

```
% Potegowanie
>> PM = A^2
PM =
     1     0     0
     8     2    -5
    14    35    -3

>> A*A
ans =
     1     0     0
     8     2    -5
    14    35    -3

>> PT = A.^2
PT =
     1     0     0
     4     9     1
     0    49     4

>> A.*A
ans =
     1     0     0
     4     9     1
     0    49     4
```

```
% Transpozycja
>> A.'  % transpozycja
ans =
    1    2    0
    0    3    7
    0   -1    2

>> A'    % hermitowskie
% sprzezenie macierzy
% (dla m. zespolonych)
ans =
    1    2    0
    0    3    7
    0   -1    2
```

Operatory macierzowe - operatory porównań

```
A = [1 0 0;
     2 3 -1;
     0 7 2] ;

B = [1 0 3;
    -1 5 2;
     2 2 2]  ;
```

```
>> A == B
ans =
   1   1   0
   0   0   0
   0   0   1

>> A ~= B
ans =
   0   0   1
   1   1   1
   1   1   0

>> 2 < A
ans =
   0   0   0
   0   1   0
   0   1   0
```

Operator	Opis
$==$	równe
$\sim=$	różne
$<$	mniejsze niż
$>$	większe niż
$<=$	mniejsze lub równe
$>=$	większe lub równe

Operatory macierzowe - operatory logiczne

```
A = [1 0 0;
     2 3 -1;
     0 7 2] ;

B = [1 0 3;
    -1 5 2;
     2 2 2]   ;
```

```
>> ~A
ans =
    0    1    1
    0    0    0
    1    0    0

>> A & B
ans =
    1    0    0
    1    1    1
    0    1    1

>> A | B
ans =
    1    0    1
    1    1    1
    1    1    1
```

Składnia

- **Instrukcja warunkowa** `if`

```
if wyrazenie_warunkowe1
instrukcje1 ;
elseif wyrazenie_warunkowe2
instrukcje2 ;
elseif
...
else
instrukcje_koncowe ;
end % endif
```

```
a = 5 ;
if ( a < 0 )
b = -1 ;
elseif a == 0
b = 0 ;
else
b = 1 ;
end
b
```

- **Instrukcja wyboru** `switch`

```
switch wyrazenie
case wartosc1
instrukcje1 ;
case wartosc2
instrukcje2 ;
...
otherwise
instrukcjen ;
end % endswitch
```

```
c = 2 ;
switch c
case 0
a = 8 ;
case 1
a = 9 ;
disp('Zmienna c ma wartosc 1') ;
otherwise
disp(['c = ' num2str(c)]) ;
end
```

Uwaga! Nie ma dwukropków i nie potrzeba break'ów!

Składnia

- **Pętla** `for`

```
for zmienna = wektor_wartosci
instrukcje
end % endfor
```

```
suma = 0 ;
w = 1:10 ;
for k = w
suma = suma + k ;
end
```

- **Pętla** `while`

```
while wyrazenie
instrukcje
end %endwhile
```

```
suma = 0 ;
k = 1 ;
while k <= 10
suma = suma + k ;
k = k + 1 ;
end
```

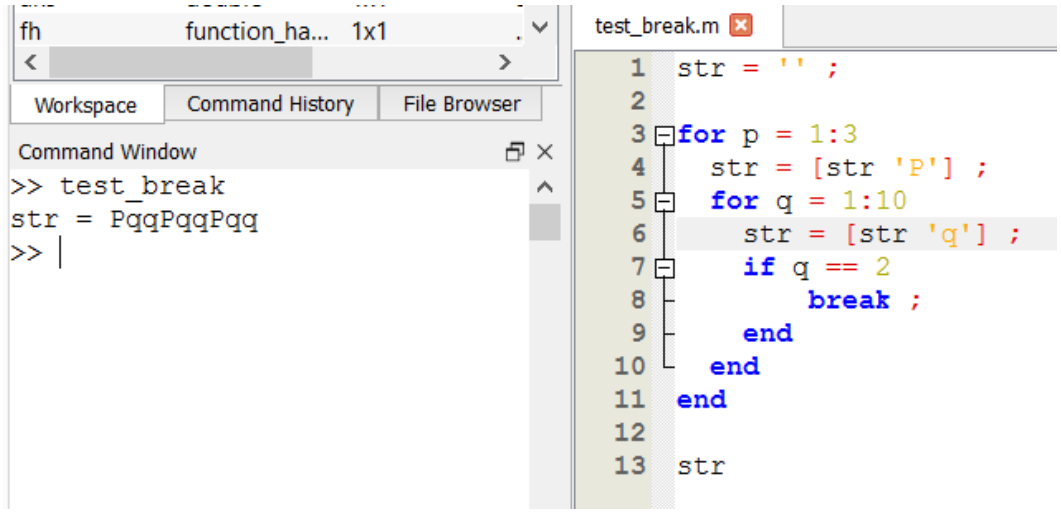
- **Pętla** `do ...until`

  Brak w Matlabie

Składnia

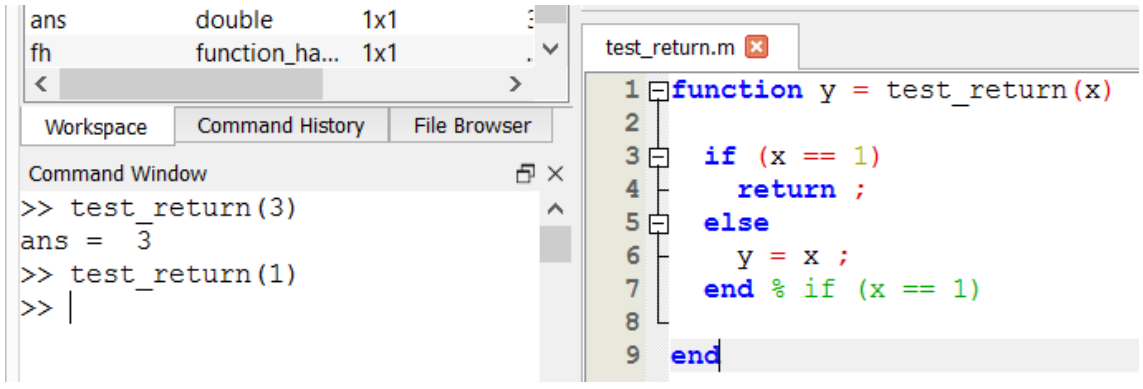
- **Instrukcja** `break`

Powoduje przerwanie wykonywania danej pętli. Opuszczany jest tylko jeden poziom zagłębienia.



- **Instrukcja** `return`

Powoduje bezwarunkowe przerwanie wykonywania danego skryptu (funkcji) i powrót do miejsca jego (jej) wywołania.



Skrypty, funkcje

- skrypty
- funkcje
  - funkcje w m-plikach

```
function [w1, w2, ... ] =
      nazwa_funkcji(p1, p2, ...)

instrukcje ;

end  % endfunction
```

```
function [k,S]=porownajIdodaj(a,b)

if (a<b)
k = -1 ;
elseif (a>b)
k = 1 ;
else
k = 0 ;
end

S = a + b ;
end
```

  - funkcje anonnimowe

```
fh = @(p1,p2, ...) ... ;
```

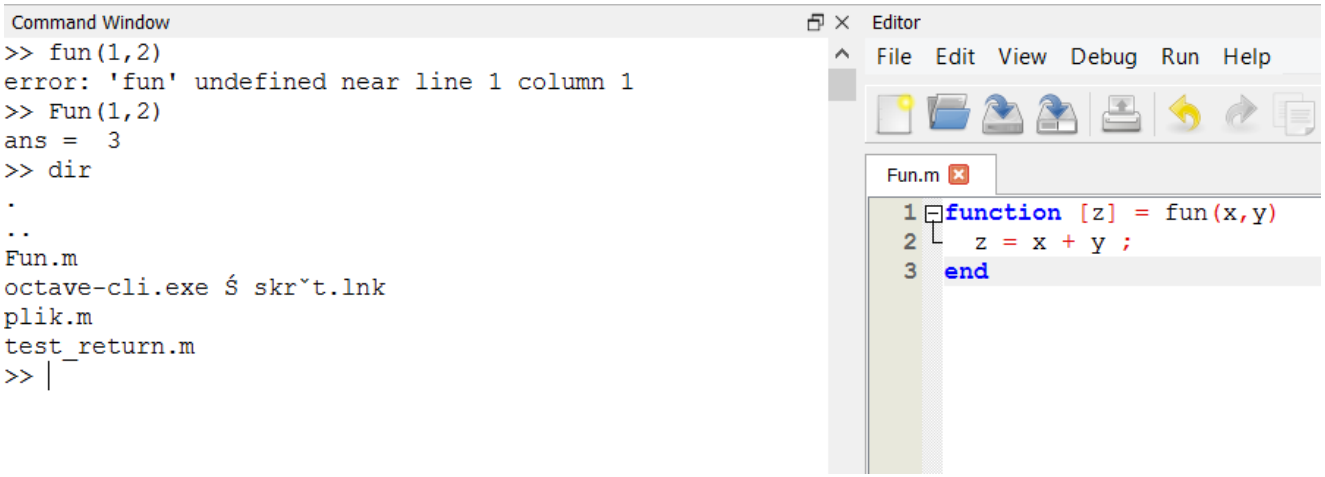
```
fh = @(x,y) 2*x + y ;
```

```
a = 2 ;
porownajIdodaj(a,6) ;
fh(a,4) ;
```

Skrypty, funkcje

Kilka zasad dobrego postępowania z funkcjami:

- nazywaj główną funkcję wewnątrz pliku tak jak sam plik
- case sensitive – wielkość ma znaczenie!



- nie używaj polskich liter, spacji, znaków operatorów arytmetycznych/logicznych w nazwie funkcji/pliku
- a jeśli miałbyś ochotę nazwać funkcję "coś" to też jej nie nazywaj "cos" bo potem nie będziesz mógł obliczyć cosinusa!
- ...oznacza, że polecenie będzie dokończone w linii poniżej

Wybrane funkcje graficzne

hold on/off
close /close all
linspace
plot
meshgrid
plot3
surf

Przydatne skróty klawiaturowe

Operator	Opis
F5	uruchomienie skryptu
F9	wykonanie zaznaczonej części kodu
Ctrl+C	przerwanie wykonywania skryptu/funkcji (przerywanie nieskończonych pętli)
Ctrl+R	zakomentowanie zaznaczonych linii
Ctrl+Shift+R	zakomentowanie zaznaczonych linii
Ctrl+Z	cofanie zmian
Ctrl+Shift+Z	cofanie ćo
Tab	blokowe wcięcie
Ctrl+Shift+Tab	cofanie blokowych wcięć
↑	poruszanie się ...
↓	... po historii poleceń

Znaki specjalne:

- średnik

```
>> a = 1 ;
>> a = 1
a =  1
```

- trzykropek

```
>> a = [1 2 ...
3 ; 4 5 6 ]

a =
    1    2    3
    4    5    6
```

- procent

```
>> a = 1 % a to jest komentarz
a =  1
```

Dziękuję za uwagę